Testing Existing Scientific Software

Presented to the ASCI V&V Working Group

Ellen Shepherd
Information Systems Engineering Center 6500

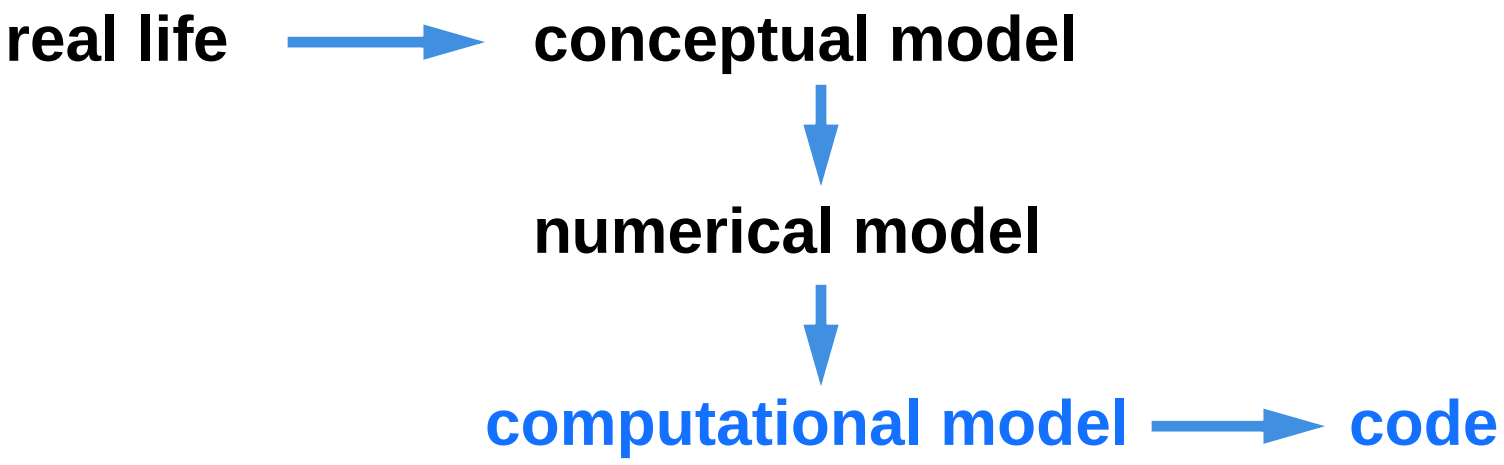
August 19, 1999

Sandia National Laboratories

Scientific software is developed in a unique environment

- Rapid prototyping often entails evolving requirements and design issues
- Existing (legacy) software is enhanced for current use
- Complex numerical models are difficult to verify and validate
- Users are very knowledgeable about use of the code
- Code may be one of many in a chain of models used for analysis
- Code is perceived as a research tool, not a formal product

There are many aspects of validation and verification in scientific software



Verification: Is the *model right?*
Validation: Is it the *right model?*

Testing verifies correctness of code and computational model

The purposes of testing are

- to establish a level of confidence
- to quantify relative improvements in successive versions
- to have a test methodology for regression testing

which are accomplished by

- testing correctness of the code
- testing code against model design
- assessing reliability

Testing provides immediate, tangible benefits

- Indicates areas for performance improvements
- Provides updates for users' manuals
- Positions software for accreditation
- Captures knowledge of domain experts
- Provides baseline for porting to other environments and platforms

Testing provides programmatic benefits

- Provides focus for application of code
- Provides focus for future development efforts
- Test efforts are readily promotable to any level of formality of process
- Test efforts are readily scalable if warranted

Testing is characterizing the behavior of the code

- Testing is not debugging
- Test for success, *and* test for failure
- Conduct conscious testing

How do I go about testing existing software?

I. Build a foundation for testing scientific software.

- **Establish and use configuration management**
- **Establish and use programming guidelines**
- **Establish a process to integrate code changes**
- **Develop a long-range, comprehensive test plan**
  (consider interdependencies of codes and data models)

II. Develop and implement test plan.

- Identify stakeholders and responsibilities.
- Obtain requirements and design specifications.
- Identify items to be tested.
- Identify features to be tested and not to be tested.
- Identify test environment(s).
- Establish test entry criteria.
- Establish pass/fail criteria.
- Have test cases reviewed and approved.
- Identify test deliverables.
- Establish schedule, identify risks and contingencies.

III. Develop appropriate test cases.

- **Establishing confidence in the use of the code requires a judicious combination of approaches**
- **Many methods support verifying correctness of implementation against design**

  static test
  dynamic test
  unit, component tests
  integration/system test
  structural test
  code review
  regression test
  test tools
  coverage analysis
  functional test

- **Test cases may come from problem domain**

  comparisons against known analytic solution, arena tests, other simulation software; sensitivity analyses; steady state comparison for transient problems

- **Test cases may focus on numerical analysis and uncertainty quantification**

  Oberkampf and DeLand, "A New Methodology for the Estimation of Total Uncertainty in Computational Simulation," AIAA-99-1612, 4/99

  Knupp and Salari, "Code Verification via the Method of Manufactured Solutions," *Draft*

- **Ensure coverage of requirements and design**

Balance costs with acceptable level of confidence

Sandia National Laboratories

Apply analogous test concepts and techniques to other components in the system

- Constituent models: Consider input and output requirements, and compatibility of assumptions across the models
- Pre- and post-processors: Verify correctness of these
- Data models: Consider the interdependence of soft-ware and data

Summary

- Testing is characterizing the behavior of the code
- Conduct conscious testing
- Test efforts have immediate, tangible benefits
- Test efforts are promotable and scalable to any level of V&V requirements

- Two excellent references on testing software:

(There may be other equally good references; here are two I've used.)

Software Testing Techniques, Boris Beizer, Paperback, 2nd edition (1990). The Coriolis Group; ISBN: 1850328803; amazon.com price $44.76.

The Art of Software Testing, Glenford Myers, Hardcover, (1979). John Wiley & Sons; ISBN: 0471043281; amazon.com price $85.00.